Q.1.

```
.dosseg
.model small
.stack 100h

.data

	arr db 5 dup(?)
	var1 db 10,13, 'Enter 4 numbers $'
	var2 db 10,13, 'Sorted array $'
.code
main proc
	mov ax,@data
	mov ds,ax

	;print add here
	mov dx,offset var1
	mov ah,9
	int 21h

	mov cx,4
	mov bx,offset arr
	mov ah,1

	inputloop:
		int 21h
		mov [bx],al
		inc bx
		loop inputloop

	mov cx,4
	dec cx

	outerloop:
		mov bx,cx
		mov si,0

	compareloop:
		mov al,arr[si]
		mov dl,arr[si+1]
		cmp al,dl

		;jnc dontswap  ; just toggle these lines for ascending and descending
		jc dontswap ; this is for asending

			mov arr[si],dl
			mov arr[si+1],al

		dontswap:
			inc si
			dec bx
			jnz compareloop

			loop outerloop

	mov ah,2
	mov dl,10
```

```
	int 21h
	mov dl,13
	int 21h

	;print sorted

	mov dx,offset var2
	mov ah,9
	int 21h

	mov cx,4
	mov bx,offset arr

	output:
		mov dl,[bx]
		mov ah,2
		int 21h

		mov dl,32
		mov ah,2
		int 21h

		inc bx
		loop output

	mov ah,4ch
	int 21h
main endp
end main
```

```
C:\>hellonew.exe

Enter 4 numbers 9876

Sorted array 6 7 8 9
C:\>
```

```
Sorted array 4 3 2 1
C:\>
```

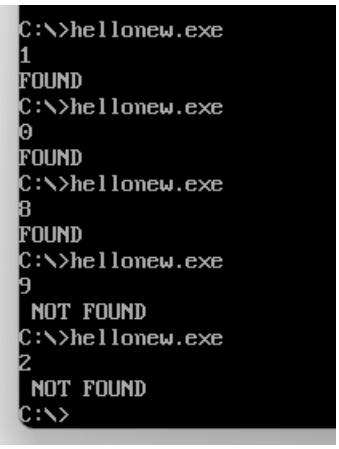
Q.2.

```
dosseg
.model small
.stack 100h
.data
array db 0,1,2,3,4,5,6,7,8,9
result db 10 Dup(?)
.code
main proc
    mov ax, @data
    mov ds, ax
    mov si, offset array
    mov di, offset result
    mov cx, 10
    l1:
        mov al, [si]
        mov bl,2
        div bl
        cmp ah, 0
        jne oddNumber ; if odd phir us ko result main dal diya

        jmp nextElement ; if even phir sirf increment kr diya

        oddNumber:
        mov al,[si]
        mov byte ptr [di], al
        inc di

        nextElement:
        inc si
    loop l1

    ; displaying the new array below
    mov si, offset result
    mov cx, 10

    l2:
        mov dl, [si]
        add dl,'0'
        mov ah, 02h
        int 21h
        inc si
    loop l2

    mov ah, 4ch
    int 21h
main endp
end main
```

```
C:\>hellonew.exe
1357900000
C:\>
```

Q.3.

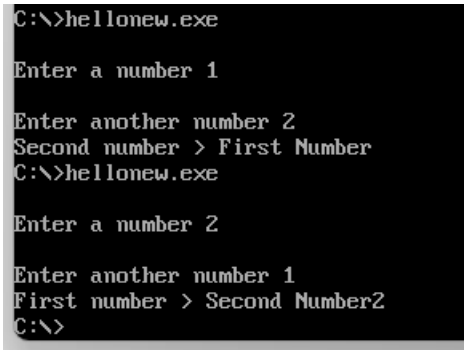
```
dosseg
.model small
.stack 100h
.data
array db '0','0','0','0','0','0','0','0','8','1'
var2 db 10,13,'FOUND $'
var3 db 10,13,' NOT FOUND $'
var1 db ?
.code
main proc
    mov ax, @data
    mov ds, ax
    mov si, offset array

    mov cx, 10

    mov ah,1
    int 21h

    mov var1,al


    l1:
        mov al, [si]

        cmp var1,al


        je cout


        inc si
        loop l1


    mov dx,offset var3
    mov ah,9
    int 21h
    jmp h

   cout:
        mov dx,offset var2
        mov ah,9
        int 21h

    h:

    mov ah, 4ch
    int 21h

main endp
end main
```



Q.4.

```
dosseg
.model small
.stack 100h
.data

    str1 db 10,13,'Enter a number $'
    str2 db 10,13,'Enter another number $'

    ans1 db 10,13,'First number > Second Number$'
    ans2 db 10,13,'Second number > First Number$'

.code

main proc
    mov ax, @data
    mov ds, ax


    mov dx,offset str1
    mov ah,09h
    int 21h



    mov ah,1
    int 21h
    mov bl,al

    mov ah, 2
    mov dl,10
    int 21h
    mov dl, 13
    int 21h

    mov dx,offset str2
    mov ah,09h
    int 21h

    mov ah,1
    int 21h
```

```
    mov bh,al

    biggernumber:
        cmp bl,bh

        jg firstgreater
        jmp here

        firstgreater:

            mov dx,offset ans1
            mov ah,09h
            int 21h

            mov ah,2
            mov dl,bl
            int 21h
            jmp exit

        here:

            mov dx,offset ans2
            mov ah,09h
            int 21h



           jmp exit

        exit:
            mov ah,4ch
            int 21h

main endp
end main
```



Q.5.

```
dosseg
.model small
.stack 100h
.data

    array db 0,0,0,0,9,0,0,0,8,1

.code
main proc
    mov ax, @data
    mov ds, ax
    mov si, offset array

    mov cx, 10

    mov bl, [si]

greaterloop:
    inc si
    cmp [si], bl

    jle greater   ;agar value choti ho to jump

    mov bl, [si]
    jmp greaterloop

greater:
    loop greaterloop


    add bl, '0'
    mov dl, bl
    mov ah, 2
    int 21h

    mov ah, 2
    mov dl,10
    int 21h
    mov dl, 13
    int 21h

    ; smaller ka kr rhy hain neechay

    mov si, offset array

    mov cx, 10

    mov bh, [si]

    smallerloop:
    inc si
    cmp [si], bh

    jge smaller

    mov bh, [si]
```

```
    jmp smallerloop

smaller:
    loop smallerloop

    add bh, '0'
    mov dl, bh
    mov ah, 2
    int 21h

    mov ah, 4ch
    int 21h

main endp
end main
```